Causality in Linear Logic

Full Completeness and Injectivity (Unit-Free Multiplicative-Additive Fragment)

Simon Castellan(✉) and Nobuko Yoshida

Imperial College London, London, UK
simon@phis.me

Abstract. Commuting conversions of Linear Logic induce a notion of dependency between rules inside a proof derivation: a rule depends on a previous rule when they cannot be permuted using the conversions. We propose a new interpretation of proofs of Linear Logic as *causal invariants* which captures *exactly* this dependency. We represent causal invariants using game semantics based on general event structures, carving out, inside the model of [6], a submodel of causal invariants. This submodel supports an interpretation of unit-free Multiplicative Additive Linear Logic with MIX (MALL$^-$) which is (1) *fully complete*: every element of the model is the denotation of a proof and (2) *injective*: equality in the model characterises exactly commuting conversions of MALL$^-$. This improves over the standard fully complete game semantics model of MALL$^-$.

Keywords: Event structures · Linear Logic · Proof nets · Game semantics

1 Introduction

Proofs up to commuting conversions. In the sequent calculus of Linear Logic, the order between rules need not always matter: allowed reorderings are expressed by *commuting conversions*. These conversions are necessary for confluence of cut-elimination by mitigating the sequentiality of the sequent calculus. The real proof object is often seen as an equivalence class of proofs modulo commuting conversions. The problem of providing a canonical representation of proofs up to those commuting conversions is as old as Linear Logic itself, and proves to be a challenging problem. The traditional solution interprets a proof by a graphical representation called *proof net* and dates back to Girard [17]. Girard's solution is only satisfactory in the multiplicative-exponential fragment of Linear Logic. For additives, a well-known solution is due to Hughes and van Glabbeck [22], where proofs are reduced to their set of axiom linkings. However, the correctness criterion relies on the difficult *toggling* condition.

Proof nets tend to be based on specific representations such as graphs or sets of linkings. Denotational semantics has not managed to provide a semantic counterpart to proof nets, which would be a model where every element is

M. Bojańczyk and A. Simpson (Eds.): FOSSACS 2019, LNCS 11425, pp. 150–168, 2019.
https://doi.org/10.1007/978-3-030-17127-8_9

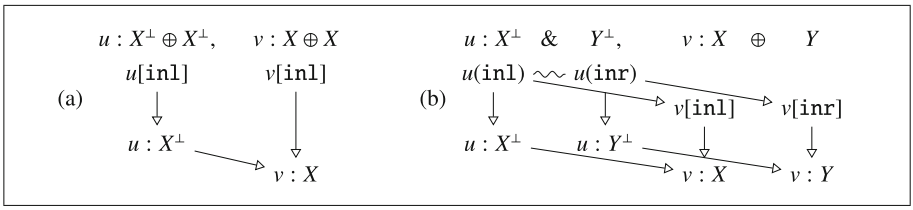


Fig. 1. Examples of causal invariants

the interpretation of a proof (*full completeness*) and whose equational theory coincides with commuting conversions (*injectivity*). We believe this is because denotational semantics views conversions as extensional principles, hence models proofs with extensional objects (relations, functions) too far from the syntax.

Conversions essentially state that the order between rules applied to different premises does not matter, as evidenced in the two equivalent proofs of the sequent $\vdash X^\perp \oplus X^\perp, X \oplus X$ depicted on the right. These two proofs are equal in extensional models of Linear Logic because *they have the same extensional behaviour*. Unfortunately, characterising the image of the interpretation proved to be a difficult task in extensional models. The first fully complete models used game semantics, and are due to Abramsky and Melliès (MALL) [1] and Melliès (Full LL) [24]. However, their models use an *extensional* quotient on strategies to satisfy the conversions, blurring the concrete nature of strategies.

$$\dfrac{\dfrac{\dfrac{}{\vdash X^\perp, X}\,Ax}{\vdash X^\perp, X \oplus X}\,\oplus_1}{\vdash X^\perp \oplus X^\perp, X \oplus X}\,\oplus_1 \qquad \dfrac{\dfrac{\dfrac{}{\vdash X^\perp, X}\,Ax}{\vdash X^\perp \oplus X^\perp, X}\,\oplus_1}{\vdash X^\perp \oplus X^\perp, X \oplus X}\,\oplus_1$$

The true concurrency of conversions. Recent work [5] highlights an interpretation of Linear Logic as communicating processes. Rules become actions whose polarity (input or output) is tied to the polarity of the connective (negative or positive), and cut-elimination becomes communication. In this interpretation, each assumption in the context is assigned a channel on which the proof communicates. Interestingly, commuting conversions can be read as asynchronous permutations. For instance, the conversion mentioned above becomes the equation in the syntax of Wadler [27]:

$$(1) \quad u[\mathtt{inl}].\, v[\mathtt{inl}].\, [u \leftrightarrow v] \equiv v[\mathtt{inl}].\, u[\mathtt{inl}].\, [u \leftrightarrow v] \;\triangleright\; u : X^\perp \oplus X^\perp, v : X \oplus X,$$

where $u[\mathtt{inl}]$ corresponds to a $\oplus_1$-introduction rule on (the assumption corresponding to) u, and $[u \leftrightarrow v]$ is the counterpart to an axiom between the hypothesis corresponding to u and v. It becomes then natural to consider that the canonical object representing these two proofs should be a concurrent process issuing the two outputs in parallel. A notion of causality emerges from this interpretation, where a rule *depends on* a previous rule below in the tree when these two rules cannot be permuted using the commuting conversions. This leads us to causal models to make this dependency explicit. For instance, the two

processes in (1) can be represented as the partial order depicted in Fig. 1a, where dependency between rules is marked with $\twoheadrightarrow$.

In presence of &, a derivation stands for several execution (slices), given by different premises of a &-rule (whose process equivalent is $u.\mathtt{case}\,(P, Q)$ and represents pattern matching on an incoming message). The identity on $X \oplus Y$, corresponding to the proof

$$u.\mathtt{case}\,(v[\mathtt{inl}].\,[u \leftrightarrow v],\quad v[\mathtt{inr}].\,[u \leftrightarrow v]) \;\triangleright\; u : X^{\perp}\,\&\,Y^{\perp}, v : X \oplus Y,$$

is interpreted by the *event structure* depicted in Fig. 1b. Event structures [28] combine a partial order, representing causality, with a conflict relation representing when two events cannot belong to the same execution (here, same slice). Conflict here is indicating with $\sim$ and separates the slices. The &-introduction becomes two conflicting events.

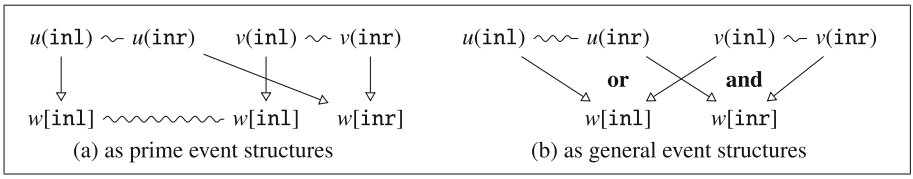


Fig. 2. Representations of `or`

Conjunctive and disjunctive causalities. Consider the process on the context $u : (X \oplus X)^{\perp}, v : (Y \oplus Y)^{\perp}, w : (X \otimes Y) \oplus (X \otimes Y)$ implementing disjunction:

$$\mathtt{or} = u.\mathtt{case}\begin{pmatrix} v.\mathtt{case}\,(w[\mathtt{inl}].\,P, w[\mathtt{inl}].\,P), \\ v.\mathtt{case}\,(w[\mathtt{inl}].\,P, w[\mathtt{inr}].\,P) \end{pmatrix} \text{ where } P = w[x].\,([u \leftrightarrow w] \mid [v \leftrightarrow x]).$$

Cuts of `or` against a proof starting with $u[\mathtt{inl}]$ or $v[\mathtt{inl}]$ answer on w after reduction:

$$(\nu u)(\mathtt{or} \mid u[\mathtt{inl}]) \to^{*} w[\mathtt{inl}].v.\mathtt{case}\,(P, P) \quad (\nu v)(\mathtt{or} \mid v[\mathtt{inl}]) \to^{*} w[\mathtt{inl}].u.\mathtt{case}\,(P, P)$$

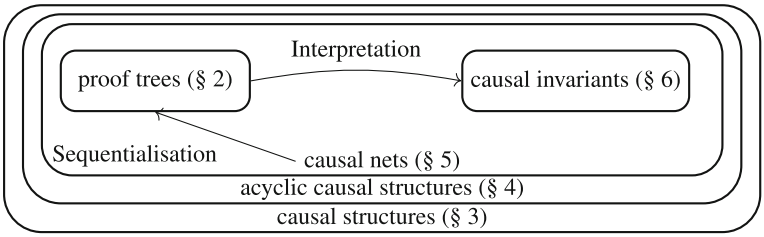
where $(\nu u)(P \mid Q)$ is the process counterpart to logical cuts. This operational behaviour is related to *parallel or*, evaluating its arguments in parallel and returning true as soon as one returns true. Due to this intentional behaviour, the interpretation of `or` in prime event structures is nondeterministic (Fig. 2a), as causality in event structures is *conjunctive* (an event may only occur after all its predecessors have occurred). By moving to *general* event structures, however, we can make the disjunctive causality explicit and recover determinism (Fig. 2b).

Contributions and outline. Drawing inspiration from the interpretation of proofs in terms of processes, we build a fully complete and injective model of unit-free Multiplicative Additive Linear Logic with MIX (MALL^{-}), interpreting proofs as general event structures living in a submodel of the model introduced by [6]. Moreover, our model captures the dependency between rules, which makes

sequentialisation a local operation, unlike in proof nets, and has a more uniform acyclicity condition than [22].

We first recall the syntax of MALL$^-$ and its reading in terms of processes in Sect. 2. Then, in Sect. 3, we present a slight variation on the model of [6], where we call the (pre)strategies *causal structures*, by analogy with proof structures. Each proof tree can be seen as a (sequential) causal structure. However, the space of causal structures is too broad and there are many causal structures which do not correspond to any proofs. A major obstacle to sequentialisation is the presence of *deadlocks*. In Sect. 4, we introduce a condition on causal structures, ensuring deadlock-free composition, inspired by the interaction between $\parr$ and $\otimes$ in Linear Logic. Acyclic causal structures are still allowed to only explore partially the game, contrary to proofs which must explore it exhaustively, hence in Sect. 5, we introduce further conditions on causal structures, ensuring a *strong* sequentialisation theorem (Theorem 2): we call them *causal nets*. In Sect. 6, we define causal invariants as maximal causal nets. Every causal net embeds in a *unique* causal invariant; and a particular proof P embeds inside a unique causal invariant which forms its denotation $[\![P]\!]$. Moreover, two proofs embed in the same causal invariant if and only if they are convertible (Theorem 4). Finally, we show how to equip causal invariants with the structure of $*$-autonomous category with products and deduce that they form a fully complete model of MALL$^-$ (Theorem 6) for which the interpretation is injective.

The proofs are available in the technical report [7].



2 MALL$^-$ and Its Commuting Conversions

In this section, we introduce MALL$^-$ formulas and proofs as well as the standard commuting conversions and cut elimination for this logic. As mentioned in the introduction, we use a process-like presentation of proofs following [27]. This highlights the communicating aspect of proofs which is an essential intuition for the model; and it offers a concise visualisation of proofs and conversions.

Formulas. We define the formulas of MALL$^-$: $T, S ::= X \mid X^\perp \mid T \otimes S \mid T \parr S \mid T \oplus S \mid T \,\&\, S$, where X and $X^\perp$ are *atomic formulas* (or *ltterals*) belonging to a set $\mathbb{A}$. Formulas come with the standard notion of duality $(\cdot)^\perp$ given by the De Morgan rules: $\otimes$ is dual to $\parr$, and $\oplus$ to $\&$. An *environment* is a partial mapping of *names* to formulas, instead of a multiset of formulas – names disambiguate which assumption a rule acts on.

Proofs as processes. We see proofs of MALL$^-$ (with MIX) as typing derivations for a variant of the π-calculus [27]. The (untyped) syntax for the processes is as follows:

$$\begin{array}{rll} P, Q ::= & u(v).\, P \mid u[v].\, (P \mid Q) & \text{(multiplicatives)} \\ \mid & u.\mathtt{case}\,(P, Q) \mid u[\mathtt{inl}].\, P \mid u[\mathtt{inr}].\, P & \text{(additives)} \\ \mid & [u \leftrightarrow v] \mid (\nu u)(P \mid Q) \mid \ (P \mid Q) & \text{(logical and mix)} \end{array}$$

$u(v).P$ denotes an input of v on channel u (used in $\parr$-introduction) while $u[v].(P \mid Q)$ denotes output of a fresh channel v along channel u (used in $\otimes$-introduction); The term $[u \leftrightarrow v]$ is a *link*, forwarding messages received on u to v, corresponds to axioms, and conversely; and $(\nu u)(P \mid Q)$ represents a restriction of u in P and Q and corresponds to cuts; $u.\mathtt{case}\,(P, Q)$ is an input branching representing &-introductions, which interacts with selection, either $u[\mathtt{inl}].\, R$ or $u[\mathtt{inr}].\, R$; in $(\nu u)(P \mid Q)$, u is bound in both P and Q, in $u(v).\, P$, v is bound in P, and in $u[v].\, (P \mid Q)$, v is only bound in Q.

We now define MALL$^-$ proofs as typing derivations for processes. The inference rules, recalled in Fig. 3, are from [27]. The links (axioms) are restricted to literals – for composite types, one can use the usual η-expansion laws. There is a straightforward bijection between standard (η-expanded) proofs of MALL$^-$ and typing derivations.

$$\frac{P \triangleright u : T, v : S, \Gamma}{u(v).\, P \triangleright u : T \parr S, \Gamma} \qquad \frac{P \triangleright u : T, \Gamma \quad Q \triangleright v : S, \Delta}{u[v].\, (P|Q) \triangleright u : T \otimes S, \Gamma, \Delta} \qquad \frac{}{[u \leftrightarrow v] \triangleright u : X^{\perp}, v : X}$$

$$\frac{P \triangleright \Gamma, u : T \quad Q \triangleright \Delta, u : T^{\perp}}{(\nu u)(P \mid Q) \triangleright \Gamma, \Delta} \qquad \frac{P \triangleright \Gamma, u : T \qquad Q \triangleright \Gamma, u : S}{u.\mathtt{case}\,(P, Q) \triangleright \Gamma, u : T \& S} \qquad \frac{P \triangleright \Gamma, u : T}{u[\mathtt{inl}].\, P \triangleright \Gamma, u : T \oplus S}$$

$$\frac{P \triangleright \Gamma, u : S}{u[\mathtt{inr}].\, P \triangleright \Gamma, u : T \oplus S} \qquad \frac{P \triangleright \Gamma \qquad Q \triangleright \Delta}{P \mid Q \triangleright \Gamma, \Delta}$$

$$\frac{}{\vdash u[\mathtt{inl}].\, [] : \Gamma, u : T \Rightarrow \Gamma, u : T \oplus S} \qquad \frac{Q \triangleright \Delta, v : S}{\vdash u[v].\, ([] \mid Q) : \Gamma, u : T \Rightarrow u : T \otimes S, \Gamma, \Delta}$$

$$\frac{}{\vdash u[\mathtt{inr}].\, [] : \Gamma, u : S \Rightarrow \Gamma, u : T \oplus S} \qquad \frac{P \triangleright \Gamma, u : T}{\vdash u[v].\, (P \mid []) : \Delta, v : S \Rightarrow u : T \otimes S, \Gamma, \Delta}$$

$$\frac{}{\vdash u.\mathtt{case}\,([]_1, []_2) : (\Gamma, u : T) \times (\Gamma, u : S) \Rightarrow \Gamma, u : T \,\&\, S}$$

$$\frac{}{\vdash u(v).\, [] : \Gamma, u : T, v : S \Rightarrow \Gamma, u : T \parr S} \qquad \frac{P \triangleright \Delta}{\vdash ([] \mid P) : \Gamma \Rightarrow \Gamma, \Delta} \qquad \frac{P \triangleright \Gamma}{\vdash (P \mid []) : \Delta \Rightarrow \Gamma, \Delta}$$

Fig. 3. Typing rules for MALL$^-$ (above) and contexts (below)

Commutation rules and cut elimination. We now explain the valid commutations rules in our calculus. We consider contexts $C\,[[]_1, \ldots, []_n]$ with several holes to accomodate & which has two branches. Contexts are defined in Fig. 3, and

are assigned a type $\Gamma_1 \times \ldots \times \Gamma_n \Rightarrow \Delta$. It intuitively means that if we plug proofs of Γ_i in the holes, we get back a proof of Δ. We use the notation $C[P_i]_i$ for $C[P_1, \ldots, P_n]$ when (P_i) is a family of processes. Commuting conversion is the smallest congruence $\equiv$ satisfying all well-typed instances of the rule $C[D[P_{i,j}]_j]_i \equiv D[C[P_{i,j}]_i]_j$ for C and D two contexts. For instance $a[\mathtt{inl}].\, b.\mathtt{case}\,(P, Q) \equiv b.\mathtt{case}\,(a[\mathtt{inl}].\, P, a[\mathtt{inl}].\, Q)$. Figure 4 gives reduction rules $P \to Q$. The first four rules are the *principal* cut rules and describe the interaction of two dual terms, while the last one allows cuts to move inside contexts.

3 Concurrent Games Based on General Event Structures

This section introduces a slight variation on the model of [6]. In Sect. 3.1, we define *games* as prime event structures with polarities, which are used to interpret formulas. We then introduce general event structures in Sect. 3.2, which are used to define causal structures.

$$(\nu u)([u \leftrightarrow v] \mid P) \to P[v/u] \qquad (\nu u)(u[x].\, (P \mid Q) \mid u(x).\, R) \to (\nu u)(P \mid (\nu x)(Q \mid R))$$

$$(\nu u)(u[\mathtt{inl}].\, R \mid u.\mathtt{case}\,(P, Q)) \to (\nu u)(R \mid P) \qquad (\nu u)(u[\mathtt{inr}].\, R \mid u.\mathtt{case}\,(P, Q)) \to (\nu u)(R \mid Q)$$

$$(\nu u)(C[P_i]_i \mid Q) \to C[(\nu u)(P_i \mid Q)]_i \quad (u \notin C)$$

Fig. 4. Cut elimination in MALL$^-$

3.1 Games as Prime Event Structures with Polarities

Definition of games. Prime event structures [28] (simply event structures in the rest of the paper) are a causal model of nondeterministic and concurrent computation. We use here prime event structures *with binary conflict*. An **event structure** is a triple $(E, \leq_E, \#_E)$ where $(E, \leq_E)$ is a partial order and $\#_E$ is an irreflexive symmetric relation (representing **conflict**) satisfying: (1) if $e \in E$, then $[e] := \{e' \in E \mid e' \leq_E e\}$ is finite; and (2) if $e \#_E e'$ and $e \leq_E e''$ then $e'' \#_E e'$. We often omit the E subscripts when clear from the context.

A **configuration** of E is a downclosed subset of E which does not contain two conflicting events. We write $\mathscr{C}(E)$ for the set of *finite* configurations of E. For any $e \in E$, $[e]$ is a configuration, and so is $(e) := [e] \setminus \{e\}$. We write $e \rightarrowtriangle e'$ for the immediate causal relation of E defined as $e < e'$ with no event between. Similarly, a conflict $e \# e'$ is **minimal**, denoted $e \sim e'$, when the $[e] \cup (e')$ and $(e) \cup [e']$ are configurations. When drawing event structures, only $\rightarrowtriangle$ and $\sim$ are represented. We write $\max(E)$ for the set of maximal events of E for $\leq_E$. An event e is maximal in x when it has no successor for $\leq_E$ in x. We write $\max_E x$ for the maximal events of a configuration $x \in \mathscr{C}(E)$.

An event structure E is **confusion-free** when (1) for all $e \sim_E e'$ then $[e] = [e')$ and (2) if $e \sim_E e'$ and $e' \sim_E e''$ then $e = e''$ or $e \sim_E e''$. As a result, the relation "$e \sim e'$ or $e = e'$" is an equivalence relation whose equivalent classes $\mathfrak{a}$ are called **cells**.

Definition 1. *A **game** is a confusion-free event structure A along with an assignment $pol : A \to \{-, +\}$ such that cells contain events of the same polarity, and a function $atom : \max(A) \to \mathbb{A}$ mapping every maximal event of A to an atom. Events with polarity $-$ (resp. $+$) are **negative** (resp. **positive**).*

Events of a game are usually called *moves*. The restriction imposes branching to be polarised (*i.e.* belonging to a player). A game is **rooted** when two minimal events are in conflict. Single types are interpreted by rooted games, while contexts are interpreted by arbitrary games. When introducing moves of a game, we will indicate their polarity in exponent, *e.g.* "let $a^+ \in A$" stands for assuming a positive move of A.

Interpretation of formulas. To interpret formulas, we make use of standard constructions on prime event structures. The event structure $a \cdot E$ is E prefixed with a, *i.e.* $E \cup \{a\}$ where *all* events of E depends on a. The parallel composition of E and E' represents parallel executions of E and E' without interference:

Definition 2. *The parallel composition of event structures A_0 and A_1 is the event structure $A_0 \parallel A_1 = (\{0\} \times A_0 \cup \{1\} \times A_1, \leq_{A_0 \parallel A_1}, \#_{A_0 \parallel A_1})$ with $(i, a) \leq_{A_0 \parallel A_1} (j, a')$ iff $i = j$ and $a \leq_{A_i} a'$; and $(i, a) \#_{A_0 \parallel A_1} (j, a')$ when $i = j$ and $a \#_{A_j} a'$.*

The sum of event structure $E + F$ is the nondeterministic analogue of parallel composition.

Definition 3. *The sum $A_0 + A_1$ of the two event structures A_0 and A_1 has the same partial order as $A_0 \parallel A_1$, and conflict relation $(i, a) \#_{A_0 + A_1} (j, a')$ iff $i \neq j$ or $i = j$ and $a \#_{A_j} a'$.*

Prefixing, parallel composition and sum of event structures extend to games. The dual of a game A, obtained by reversing the polarity labelling, is written $A^\perp$. Given $x \in \mathscr{C}(A)$, we define A/x ("A after x") as the subgame of A comprising the events $a \in A \setminus x$ not in conflict with events in x.

Interpretation of formulas. The interpretation of the atom X is the game with a single positive event simply written X with $atom(X) = X$, and the interpretation of $X^\perp$ is $[\![X]\!]^\perp$, written simply $X^\perp$ in diagrams. For composite formulas, we let (where `send`, `inl` and `inr` are simply labels):

$$[\![S \otimes T]\!] = \texttt{send}^+ \cdot ([\![S]\!] \parallel [\![T]\!]) \qquad [\![S \parr T]\!] = \texttt{send}^- \cdot ([\![S]\!] \parallel [\![T]\!])$$
$$[\![S \oplus T]\!] = (\texttt{inl}^+ \cdot [\![S]\!]) + (\texttt{inr}^+ \cdot [\![T]\!]) \qquad [\![S \,\&\, T]\!] = (\texttt{inl}^- \cdot [\![S]\!]) + (\texttt{inr}^- \cdot [\![T]\!])$$

Parallel composition is used to interpret contexts: $[\![u_1 : T_1, \ldots, u_n : T_n]\!] = [\![T_1]\!] \parallel \ldots \parallel [\![T_n]\!]$. The interpretation commutes with duality: $[\![T]\!]^\perp = [\![T^\perp]\!]$.

In diagrams, we write moves of a context following the syntactic convention: for instance $u[\texttt{inl}]$ denotes the minimal `inl` move of the u component. For tensors and pars, we use the notation $u[v]$ and $u(v)$ to make explicit the variables we use in the rest of the diagram, instead of $\texttt{send}^+$ and $\texttt{send}^-$ respectively. For atoms, we use $u : X$ and $u : X^\perp$.

3.2 Causal Structures as Deterministic General Event Structures

As we discussed in Sect. 1, prime event structures cannot express disjunctive causalities deterministically, hence fail to account for the determinism of LL. Our notion of causal structure is based on *general event structures*, which allow more complex causal patterns. We use a slight variation on the definition of deterministic general event structures given by [6], to ensure that composition is well-defined without further assumptions.

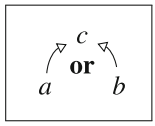
Instead of using the more concrete representation of general event structures in terms of a set of events and an enabling relation, we use the following formulation in terms of set of configurations, more adequate for mathematical reasoning. Being only sets of configurations, they can be reasoned on with very simple set-theoretic arguments.

Definition 4. *A **causal structure** (abbreviated as causal struct) on a game A is a subset $\sigma \subseteq \mathscr{C}(A)$ containing $\emptyset$ and satisfying the following conditions:*

Coincidence-freeness *If $e, e' \in x \in \sigma$ then there exists $y \in \sigma$ with $y \subseteq x$ and $y \cap \{e, e'\}$ is a singleton.*

Determinism *for $x, y \in \sigma$ such that $x \cup y$ does not contain any minimal negative conflict, then $x \cup y \in \sigma$.*

Configurations of prime event structures satisfy a further axiom, *stability*, which ensures the absence of disjunctive causalities. When σ is a causal struct on A, we write $\sigma : A$. We draw as regular event structures, using $\rightarrowtriangle$ and $\frown$. To indicate disjunctive causalities, we annotate joins with **or**. This convention is not powerful enough to draw *all* causal structs, but enough for the examples in this paper. As an example, on $A = a \parallel b \parallel c$ the diagram on the right denotes the following causal struct $\sigma = \{x \in \mathscr{C}(A) \mid c \in x \Rightarrow x \cap \{a, b\} \neq \emptyset\}$.



A **minimal event** of $\sigma : A$ is an event $a \in A$ with $\{a\} \in \sigma$. An event $a \in x \in \sigma$ is **maximal** in x when $x \setminus \{a\} \in \sigma$. A **prime configuration** of $a \in A$ is a configuration $x \in \sigma$ such that a is its unique maximal event. Because of disjunctive causalities, an event $a \in A$ can have several distinct prime configurations in σ (unlike in event structures). In the previous example, since c can be caused by either a or b, it has two prime configurations: $\{a, c\}$ and $\{b, c\}$. We write $\max \sigma$ for the set of **maximal configurations** of σ, ie. those configurations that cannot be further extended.

Even though causality is less clear in general event structures than in prime event structures, we give here a notion of immediate causal dependence that will be central to define acyclic causal structs. Given a causal struct $\sigma : A$ and $x \in \sigma$, we define a relation $\rightarrowtriangle_{x,\sigma}$ on x as follows: $a \rightarrowtriangle_{x,\sigma} a'$ when there exists a prime configuration y of a' such that $x \cup y \in \sigma$, and that a is maximal in $y \setminus \{a'\}$. This notion is compatible with the drawing above: we have $a \rightarrowtriangle_{\emptyset} c$ and $b \rightarrowtriangle_{\emptyset} c$ as c has two prime configurations: $\{a, c\}$ and $\{b, c\}$. Causality needs to be contextual, since different slices can implement different causal patterns. Parallel composition and prefixing structures extend to causal structs:

$$\sigma \parallel \tau = \{x \parallel y \in \mathscr{C}(A \parallel B) \mid (x, y) \in \sigma \times \tau\} \qquad a \cdot \sigma = \{x \in \mathscr{C}(a \cdot A) \mid x \cap A \in \sigma\}.$$

Categorical setting. Causal structs can be composed using the definitions of [6]. Consider $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$. A **synchronised configuration** is a configuration $x \in \mathscr{C}(A \parallel B \parallel C)$ such that $x \cap (A \parallel B) \in \sigma$ and $x \cap (B \parallel C) \in \tau$. A synchronised configuration x is **reachable** when there exists a sequence (**covering chain**) of synchronised configurations $x_0 = \emptyset \subseteq x_1 \subseteq \ldots \subseteq x_n = x$ such that $x_{i+1} \setminus x_i$ is a singleton. The reachable configurations are used to define the interaction $\tau \circledast \sigma$, and then after hiding, the composition $\tau \odot \sigma$:

$$\tau \circledast \sigma = \{x \text{ is a reachable synchronised configuration}\} \quad \tau \odot \sigma = \{x \cap (A \parallel C) \mid x \in \tau \circledast \sigma\}.$$

Unlike in [6], our determinism is strong enough for $\tau \odot \sigma$ to be a causal struct.

Lemma 1. *If $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ are causal structs then $\tau \odot \sigma$ is a causal struct.*

Composition of causal structs will be used to interpret cuts between proofs of Linear Logic. In concurrent game semantics, composition has a natural identity, asynchronous copycat [25], playing on the game $A^\perp \parallel A$, forwarding negative moves on one side to the positive occurrence on the other side. Following [6], we define $\mathfrak{a}_A = \{x \parallel y \in \mathscr{C}(A^\perp \parallel A) \mid y \supseteq^-_A x \cap y \subseteq^+_A x\}$ where $x \subseteq^p y$ means $x \subseteq y$ and $pol(y \setminus x) \subseteq \{p\}$.

However, in general copycat is not an identity on all causal structs, only $\sigma \subseteq \mathfrak{a}_A \odot \sigma$ holds. Indeed, copycat represents an asynchronous buffer, and causal structs which expects messages to be transmitted synchronously may be affected by composition with copycat. We call causal structs that satisfy the equality **asynchronous**. From [6], we know that asynchronous causal structs form a compact-closed category.

The syntactic tree. The syntactic tree of a derivation $P \triangleright \Delta$ can be read as a causal struct $Tr(P)$ on $[\![\Delta]\!]$, which will be the basis for our interpretation. It is defined by induction:

$$Tr(u(v).\,P) = u(v) \cdot Tr(P) \qquad Tr(u[v].\,(P \mid Q)) = u[v] \cdot (Tr(P) \parallel Tr(Q))$$
$$Tr(a.\mathtt{case}\,(P, Q)) = (a(\mathtt{inl}) \cdot Tr(P)) \cup (a(\mathtt{inr}) \cdot Tr(Q))$$
$$Tr(a[\mathtt{inl}].\,P) = a[\mathtt{inl}] \cdot Tr(P) \qquad Tr(a[\mathtt{inr}].\,P) = a[\mathtt{inr}] \cdot Tr(P)$$
$$Tr([a \leftrightarrow b]) = \mathfrak{a}_{[\![X]\!]} \text{ where } \Delta = a : X^\perp, b : X \qquad Tr(P \mid Q) = Tr(P) \parallel Tr(Q)$$
$$Tr((\nu a)(P \mid Q)) = Tr(P) \odot Tr(Q)$$

We use the convention in the diagram, for instance $u[v]$ means the initial `send` move of the u component. An example of this construction is given in Fig. 5a. Note that it is not asynchronous.

4 Acyclicity of Causal Structures

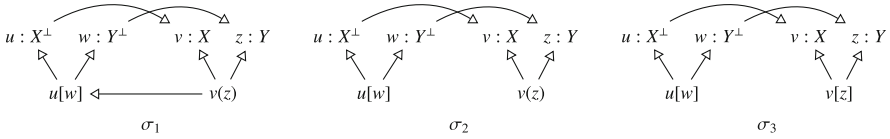
The space of causal structs is unfortunately too broad to provide a notion of causal nets, due in particular to the presence of deadlocks during composition.

As a first step towards defining causal nets, we introduce in this section a condition on causal structs inspired by the tensor rule in Linear Logic. In Sect. 4.1, we propose a notion of communication between actions, based on causality. In Sect. 4.2, we introduce a notion of acyclicity which is shown to be stable under composition and ensure deadlock-free composition.

4.1 Communication in Causal Structures

The tensor rule of Linear Logic says that after a tensor $u[v]$, the proof splits into two independent subproofs, one handling u and the other v. This syntactic condition is there to ensure that there are no communications between u and v. More precisely, we want to prevent any dependence between subsequent actions on u and an action v. Indeed such a causal dependence could create a deadlock when facing a par rule $u(v)$, which is allowed to put arbitrary dependence between such subsequent actions.

Communication in MLL. Let us start by the case of MLL, which corresponds to the case where games do not have conflicts. Consider the following three causal structs:



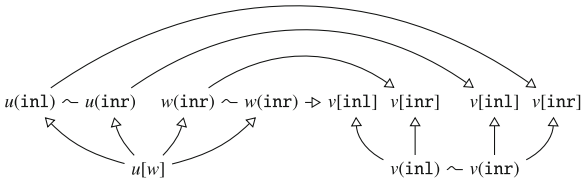
The causal structs σ_1 and σ_2 play on the game $[\![u : X^\perp \otimes Y^\perp, v : X \parr Y]\!]$, while σ_3 plays on the game $[\![u : X^\perp \otimes Y^\perp, v : X \otimes Y]\!]$. The causal structs σ_2 and σ_3 are very close to proof nets, and it is easy to see that σ_2 represents a correct proof net while σ_3 does not. In particular, there exists a proof P such that $Tr(P) \subseteq \sigma_2$ but there are no such proof Q for σ_3. Clearly, σ_3 should not be acyclic. But should σ_2? After all it is sequentialisable. But, in all sequentialisations of σ_2, the par rule $v(z)$ is applied *before* the tensor $u[w]$, and this dependency is not reflected by σ_2. Since our goal is exactly to compute these implicit dependencies, we will only consider σ_1 to be acyclic, by using a stronger sequentialisation criterion:

Definition 5. *A causal struct $\sigma : [\![\Gamma]\!]$ is **strongly sequentialisable** when for all $x \in \sigma$, there exists $P \triangleright \Gamma$ with $x \in Tr(P)$ and $Tr(P) \subseteq \sigma$.*

To understand the difference between σ_1 and σ_2, we need to look at causal chains. In both σ_1 and σ_2, we can go from $u : X^\perp$ to $w : Y^\perp$ by following immediate causal links $\rightarrowtriangle$ in any direction, but observe that in σ_1 they must all cross an event below $u[w]$ (namely $v(z)$ or $u[w]$). This prompts us to define a notion of communication *outside a configuration* x:

Definition 6. *Given $\sigma : A$ and $x \in \sigma$ we say that $a, a' \in A \setminus x$ communicate* outside *x (written $a \leftrightsquigarrow_{x,\sigma} a'$) when there exists a chain $a \leftrightarrowtriangle_{x,\sigma} a_0 \leftrightarrowtriangle_\sigma \cdots \leftrightarrowtriangle_{x,\sigma} a_n \leftrightarrowtriangle_\sigma a'$ where all the $a_i \in A \setminus x$, and $\leftrightarrowtriangle_{x,\sigma}$ denotes the symmetric closure of $\rightarrowtriangle_{x,\sigma}$.*

Communication in MALL. In presence of additives, immediate causality is not the only vector of communication. Consider the following causal struct σ_4, playing on the context $u : (A \,\&\, A) \otimes (A \,\&\, A), v : (A \oplus A) \,\&\, (A \oplus A)$ where A is irrelevant:



This pattern is not strongly sequentialisable: the tensor $u[w]$ must always go after the &-introduction on v, since we need this information to know how whether v should go with u or w when splitting the context. Yet, it is not possible to find a communication path from one side to the other by following purely causal links without crossing $u[w]$. There is however a path that uses both immediate causality and *minimal conflict*. This means that we should identify events in minimal conflict, since they represent the same (&-introduction rule). Concretely, this means lifting the previous definition at the level of cells. Given an causal struct $\sigma : A$ and $x \in \sigma$, along with two cells $\mathfrak{a}, \mathfrak{a}'$ of A/x, we define the relation $\mathfrak{a} \leftrightarrow_{x,\sigma} \mathfrak{a}'$ when there exists $a \in \mathfrak{a}$ and $a' \in \mathfrak{a}'$ such that $a \leftrightarrow_{x,\sigma} a'$; and $\mathfrak{a} \rightsquigarrow_{x,\sigma} \mathfrak{a}'$ when there exists $\mathfrak{a} \leftrightarrow_{x,\sigma} \mathfrak{a}_0 \leftrightarrow_{x,\sigma} \cdots \leftrightarrow_{\sigma} \mathfrak{a}_n \leftrightarrow_{\sigma} \mathfrak{a}'$ where all the $\mathfrak{a}_i$ do not intersect x. For instance, the two cells which are successors of the tensor $u[w]$ in σ_4 communicate outside the configuration $\{u[w]\}$ by going through the cell $\{v(\mathtt{inl}), v(\mathtt{inr})\}$.

4.2 Definition of Acyclicity on Casual Structures

Since games are trees, two events a, a' are either incomparable or have a meet $a \wedge a'$. If $a \wedge a'$ is defined and positive, we say that a and a' **have positive meet**, and means that they are on two distinct branches of a tensor. If $a \wedge a'$ is undefined, or defined and negative, we say that $a \wedge a'$ has a **negative meet**. When the meet is undefined, it means that a and a' are events of different components of the context. We consider the meet to be negative in this case, since components of a context are related by an implicit par.

These definitions are easily extended to cells. The meet $\mathfrak{a} \wedge \mathfrak{a}'$ of two cells $\mathfrak{a}$ and $\mathfrak{a}'$ of A is the meet $a \wedge a'$ for $a \in \mathfrak{a}$ and $a' \in \mathfrak{a}'$: by confusion-freeness, it does not matter which ones are chosen. Similarly, we say that $\mathfrak{a}$ and $\mathfrak{a}'$ have positive meet if $\mathfrak{a} \wedge \mathfrak{a}'$ is defined and positive; and have negative meet otherwise. These definitions formalise the idea of "the two sides of a tensor", and allow us to define acyclicity.

Definition 7. *A causal struct $\sigma : A$ is **acyclic** when for all $x \in \sigma$, for any cells $\mathfrak{a}, \mathfrak{a}'$ not intersecting x and with positive meet, if $\mathfrak{a} \rightsquigarrow_{x,\sigma} \mathfrak{a}'$ then $\mathfrak{a} \wedge \mathfrak{a}' \notin x$.*

This captures the desired intuition: if $\mathfrak{a}$ and $\mathfrak{a}'$ are on two sides of a tensor a (ie. have positive meet), and there is a communication path outside x relating them,

then a must also be outside x (and implicitly, the communication path must be going through a).

Reasoning on the interaction of acyclic strategies proved to be challenging. We prove that acyclic strategies compose, and their interaction are deadlock-free, when composition is on a rooted game B. This crucial assumption arises from the fact that in linear logic, cuts are on *formulas*. It entails that for any $b, b' \in B$, $b \wedge b'$ is defined, hence must be positive either from the point of view of σ or of τ.

Theorem 1. *For acyclic causal structs $\sigma : A^{\perp} \parallel B$ and $\tau : B^{\perp} \parallel C$, (1) their interaction is* deadlock-free*: $\tau \circledast \sigma = (\sigma \parallel C) \cap (A \parallel \tau)$; and (2) the causal struct $\tau \odot \sigma$ is acyclic.*

As a result, acyclic and asynchronous causal structs form a category. We believe this intermediate category is interesting in its own right since it generalises the deadlock-freeness argument of Linear Logic without having to assume other constraints coming from Linear Logic, such as linearity. In the next section, we study further restriction on acyclic causal structs which guarantee strong sequentialisability.

5 Causal Nets and Sequentialisation

We now ready to introduce causal nets. In Sect. 5.1, we give their definition by restricting acyclic causal structs and in Sect. 5.2 we prove that causal nets are strongly sequentialisable.

5.1 Causal Nets: Totality and Well-Linking Casual Structs

To ensure that our causal structs are strongly sequentialisable, acyclicity is not enough. First, we need to require causal structs to respect the linearity discipline of Linear Logic:

Definition 8. *A causal struct $\sigma : A$ is **total** when* (1) *for $x \in \sigma$, if x is maximal in σ, then it is maximal in $\mathscr{C}(A)$; and* (2) *for $x \in \sigma$ and $a^- \in A \setminus x$ such that $x \cup \{a\} \in \sigma$, then whenever $a \frown_A a'$, we also have $x \cup \{a'\} \in \sigma$ as well.*

The first condition forces a causal struct to play until there are no moves to play, and the second forces an causal struct to be receptive to all Opponent choices, not a subset.

Our last condition constrains axiom links. A **linking** of a game A is a pair (x, ℓ) of a $x \in \max \mathscr{C}(A)$, and a bijection $\ell : (\max_A x)^- \simeq (\max_A x)^+$ preserving the *atom* labelling.

Definition 9. *A total causal struct $\sigma : A$ is **well-linking** when for each $x \in \max(\sigma)$, there exists a linking ℓ_x of x, such that if y is a prime configuration of $\ell_x(e)$ in x, then $\max(y \setminus \{\ell_x(e)\}) = \{e\}$.*

This ensures that every positive atom has a unique predecessor which is a negative atom.

Definition 10. *A **causal net** is an acyclic, total and well-linking causal struct.*

A causal net $\sigma : A$ induces a set of linkings A, $\mathsf{link}(\sigma) := \{\ell_x \mid x \in \max \sigma\}$. The mapping $\mathsf{link}(\cdot)$ maps causal nets to the proof nets of [22].

5.2 Strong Sequentialisation of Causal Nets

Our proof of sequentialisation relies on an induction on causal nets. To this end, we provide an inductive deconstruction of parallel proofs. Consider $\sigma : A$ a causal net and a minimal event $a \in \sigma$ not an atom. We write A/a for $A/\{a\}$. Observe that if $A = [\![\Delta]\!]$, it is easy to see that there exists a context Δ/a such that $[\![\Delta/a]\!] \cong A/a$. Given a causal struct $\sigma : A$, we define the causal struct $\sigma/a = \{x \in \mathscr{C}(A/a) \mid x \cup \{a\} \in \sigma\} : A/a$.

Lemma 2. *σ/a is a causal net on A/a.*

When a is positive, we can further decompose σ/a in disjoint parts thanks to acyclicity. Write $\mathfrak{a}_1, \ldots, \mathfrak{a}_n$ for the minimal cells of A/a and consider for $n \geq k > 0$, $A_k = \{a' \in A/a \mid \mathrm{cell}(a') \leftrightsquigarrow_{\{a\},\sigma} \mathfrak{a}_k\}$. A_k contains the events of A/a which σ connects to the k-th successor of a. We also define the set $A_0 = A/a \backslash \bigcup_{1 \leq k \leq n} A_k$, of events not connected to any successor of a (this can happen with MIX). It inherits a game structure from A.

Each subset inherits a game structure from A/a. By acyclicity of σ, the A_k are pairwise disjoint, so $A/a \cong A_0 \parallel \ldots \parallel A_n$. For $0 \leq k \leq n$, define $\sigma_k = \mathscr{C}(A_k) \cap \sigma/a$.

Lemma 3. *σ_k is a causal net on A_k and we have $\sigma/a = \sigma_0 \parallel \ldots \parallel \sigma_n$.*

This formalises the intuition that after a tensor, an acyclic causal net must be a parallel composition of proofs (following the syntactic shape of the tensor rule of Linear Logic). From this result, we show by induction that any causal net is strongly sequentialisable.

Theorem 2. *If $\sigma : A$ is a causal net, then σ is strongly sequentialisable.*

We believe sequentialisation without MIX requires causal nets to be *connected*: two cells with negative meets always communicate outside any configuration they are absent from. We leave this lead for future work.

6 Causal Invariants and Completeness

Causal nets are naturally ordered by inclusion. When $\sigma \subseteq \tau$, we can regard τ as a less sequential implementation of σ. Two causal nets which are upper bounded by a causal net should represent the same proof, but with varying degrees of sequentiality. Causal nets which are maximal for inclusion (among causal nets) are hence most parallel implementations of a certain behaviour and capture our intuition of causal invariants.

Definition 11. *A **causal invariant** is a causal net $\sigma : A$ maximal for inclusion.*

6.1 Causal Invariants as Maximal Causal Nets

We start by characterising when two causal nets are upper-bounded for inclusion:

Proposition 1. *Given two causal nets $\sigma, \tau : A$, the following are equivalent:*

1. *there exists a causal net $\upsilon : A$ such that $\sigma \subseteq \upsilon$ and $\tau \subseteq \upsilon$,*
2. *the set $\sigma \vee \tau = \{x \cup y \mid x \in \sigma, y \in \tau, x \cup y \in \mathscr{C}(A)\}$ is a causal net on A,*
3. *$\mathsf{link}(\sigma) = \mathsf{link}(\tau)$.*

In this case we write $\sigma \uparrow \tau$ and $\sigma \vee \tau$ is the least upper bound of σ and τ for $\subseteq$.

It is a direct consequence of Proposition 1 that any causal net σ is included in a unique causal invariant $\sigma^{\uparrow} : A$, defined as: $\sigma^{\uparrow} = \bigvee_{\sigma \subseteq \tau} \tau$, where τ ranges over causal nets.

Lemma 4. *For $\sigma, \tau : A$ causal nets, $\sigma \uparrow \tau$ iff $\sigma^{\uparrow} = \tau^{\uparrow}$. Moreover, if σ and τ are causal invariants, $\sigma \uparrow \tau$ if and only if $\sigma = \tau$.*

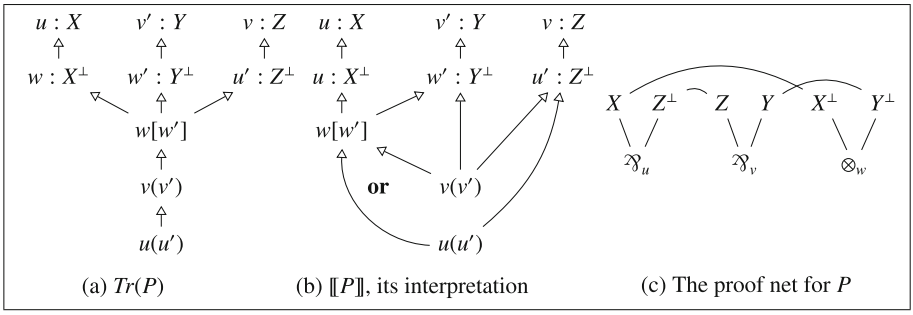


Fig. 5. Interpreting $P = u(u').v(v').w[w'].([u \leftrightarrow w] \mid ([w' \leftrightarrow v'] \mid [u' \leftrightarrow v]))$ in the context $u : X \parr Z^{\perp}, v : Z \parr Y, w : X^{\perp} \otimes Y^{\perp}$

The interpretation of a proof $P \triangleright \Delta$ is simply defined as $[\![P]\!] = Tr(P)^{\uparrow}$. Figure 5c illustrates the construction on a proof of MLL+mix. The interpretation features a disjunctive causality, as the tensor can be introduced as soon as *one* of the two pars has been.

Defining $\mathsf{link}(P) = \mathsf{link}(Tr(P))$, we have from Lemma 4: $\mathsf{link}(P) = \mathsf{link}(Q)$ if and only if $[\![P]\!] = [\![Q]\!]$. This implies that our model has the same equational theory than the proof nets of [22]. Such proof nets are already complete:

Theorem 3 ([22]). *For P, Q two proofs of Γ, we have $P \equiv Q$ iff $\mathsf{link}(P) = \mathsf{link}(Q)$.*

As a corollary, we get:

Theorem 4. *For cut-free proofs P, Q we have $P \equiv Q$ iff $[\![P]\!] = [\![Q]\!]$.*

The technical report [7] also provides an inductive proof not using the result of [22]. A consequence of this result, along with *strong* sequentialisation is: $[\![P]\!] = \bigcup_{Q \equiv P} Tr(Q)$. This equality justifies our terminology of "causal completeness", as for instance it implies that the minimal events of $[\![P]\!]$ correspond exactly the possible rules in P that can be pushed to the front using the commuting conversions.

6.2 The Category of Causal Invariants

So far we have focused on the static. Can we integrate the dynamic aspect of proofs as well? In this section, we show that causal invariants organise themselves in a category. First, we show that causal nets are stable under composition:

Lemma 5. *If $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ are causal nets, then so is $\tau \odot \sigma$.*

Note that totality requires acyclicity (and deadlock-freedom) to be stable under composition. However, causal invariants are not stable under composition: $\tau \odot \sigma$ might not be maximal, even if τ and σ are. Indeed, during the interaction, some branches of τ will not be explored by σ and vice-versa which can lead to new allowed reorderings. However, we can always embed $\tau \odot \sigma$ into $(\tau \odot \sigma)^\uparrow$:

Lemma 6. *Rooted games and causal invariants form a category **CInv**, where the composition of $\sigma : A^\perp \parallel B$ and $\tau : B^\perp \parallel C$ is $(\tau \odot \sigma)^\uparrow$ and the identity on A is $\mathfrak{a}_A^\uparrow$.*

Note that the empty game is an object of **CInv**, as we need a monoidal unit.

Monoidal-closed structure. Given two games A and B we define $A \otimes B$ as $\mathtt{send}^+ \cdot (A \parallel B)$, and 1 as the empty game. There is an obvious isomorphism $A \otimes 1 \cong A$ and $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$ in **CInv**. We now show how to compute directly the functorial action of $\otimes$, without resorting to $^\uparrow$. Consider $\sigma \in \mathbf{CInv}(A, B)$ and $\tau \in \mathbf{CInv}(C, D)$. Given $x \in \mathscr{C}((A \otimes C)^\perp \parallel (B \otimes D))$, we define $x\langle\sigma\rangle = x \cap (A^\perp \parallel B)$ and $x\langle\tau\rangle = x \cap (C^\perp \parallel D)$. If $x\langle\sigma\rangle \in \sigma$ and $x\langle\tau\rangle \in \tau$, we say that x is connected when there exists cells $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ and $\mathfrak{d}$ of A, B, C and D respectively such that $\mathfrak{a} \leftrightsquigarrow_{x\langle\sigma\rangle,\sigma} \mathfrak{c}$ and $\mathfrak{b} \leftrightsquigarrow_{x\langle\tau\rangle,\tau} \mathfrak{d}$. We define:

$$\sigma \otimes \tau = \left\{ \begin{array}{l} x \in \mathscr{C}((A \otimes C)^\perp \parallel (B \otimes D)) \text{ such that :} \\ \quad (1)\ x\langle\sigma\rangle \in \sigma \text{ and } x\langle\tau\rangle \in \tau \\ \quad (2) \text{ if } x \text{ is connected and contains } \mathtt{send}^+, \text{ then } \mathtt{send}^- \in x \end{array} \right\}$$

In (2), $\mathtt{send}^-$ refers to the minimal move of $(A \otimes C)^\perp$ and $\mathtt{send}^+$ to the one of $B \otimes D$. (2) ensures that $\sigma \otimes \tau$ is acyclic.

Lemma 7. *The tensor product defines a symmetric monoidal structure on **CInv**.*

Define $A \parr B = (A^\perp \otimes B^\perp)^\perp$, $\perp = 1 = \emptyset$ and $A \multimap B = A^\perp \parr B$.

Lemma 8. *We have a bijection $\parr_{B,C}$ between causal invariants on $A \parallel B \parallel C$ and on $A \parallel (B \parr C)$. As a result, there is an adjunction $A \otimes \_ \dashv A \multimap \_$.*

Lemma 8 implies that $\mathbf{CInv}((A \multimap \bot) \multimap \bot) \simeq \mathbf{CInv}(A)$, and **CInv** is $*$-autonomous.

Cartesian products. Given two games A, B in **CInv**, we define their product $A \,\&\, B = \mathtt{inl}^- \cdot A + \mathtt{inr}^- \cdot B$. We show how to construct the pairing of two causal invariants concretely. Given $\sigma \in \mathbf{CInv}(A, B)$ and $\tau \in \mathbf{CInv}(A, C)$, we define the common behaviour of σ and τ on A to be those $x \in \mathscr{C}(A^\perp) \cap \sigma \cap \tau$ such that for all $\mathfrak{a}, \mathfrak{a}'$ outside of x with positive meet, $\mathfrak{a} \leftrightsquigarrow_{x,\sigma} \mathfrak{a}'$ iff $\mathfrak{a} \leftrightsquigarrow_{x,\tau} \mathfrak{a}'$. We write $\sigma \cap_A \tau$ for the set of common behaviours of σ and τ and define: $\langle \sigma, \tau \rangle = (L^- \cdot \sigma) \cup (R^- \cdot \tau) \cup \sigma \cap_A \tau$. The projections are defined using copycat: $\pi_1 = \{x \in \mathscr{C}((A \,\&\, B)^\perp \parallel A) \mid x \cap (A^\perp \parallel A) \in \mathrm{cc}_A^\uparrow\}$ (and similarly for π_2).

Theorem 5. ***CInv*** *has products. As it is also $*$-autonomous, it is a model of MALL.*

It is easy to see that the interpretation of MALL$^-$ in **CInv** following the structure is the same as $[\![\cdot]\!]$, however it is computed compositionally without resorting to the $^\uparrow$ operator. We deduce that our interpretation is invariant by cut-elimination: if $P \to Q$, then $[\![P]\!] = [\![Q]\!]$. Putting the pieces together, we get the final result.

Theorem 6. ***CInv*** *is an injective and fully complete model of MALL$^-$.*

7 Extensions and Related Work

The model provides a representation of proofs which retains only the necessary sequentiality. We study the phenomenon in Linear Logic, but commuting conversions of additives arise in other languages, eg. in functional languages with sums and products, where proof nets do not necessarily exist. Having an abstract representation of which reorderings are allowed could prove useful (reasoning on the possible commuting conversions in a language with sum types is notoriously difficult).

Extensions. Exponentials are difficult to add, as their conversions are not as canonical as those of MALL. Cyclic proofs [2] could be accomodated via recursive event structures.

Adding multiplicative units while keep determinism is difficult, as their commuting conversion is subtle (*e.g.* conversion for MLL is PSPACE-complete [18]), and exhibit apparent nondeterminism. For instance the following proofs are convertible in MLL:

$$a().\, b[] \mid c[] \equiv a().\, (b[] \mid c[]) \equiv b[] \mid a().\, c[] \triangleright a : \bot, b : 1, c : 1$$

where $a().\, P$ is the process counterpart to introduction of $\bot$ and $a[]$ of 1. Intuitively, $b[]$ and $c[]$ can be performed at the start, but as soon as one is performed,

the other has to wait for the input on a. This cannot be modelled inside deterministic general event structures, as it is only deterministic against an environment that will emit on b. In contrast, proofs of MALL$^-$ remain deterministic even if their environment is not total.

We would also be interested in recast multifocusing [9] in our setting by defining a class of focussed causal nets, where there are no concurrency between positive and negative events, and show that sequentialisation always give a focused proof.

Related work. The first fully complete model of MALL$^-$ is based on closure operators [1], later extended to full Linear Logic [24]. True concurrency is used to define innocence, on which the full completeness result rests. However their model does not take advantage of concurrency to account for permutations, as strategies are sequential. This investigation has been extended to concurrent strategies by Mimram and Melliès [25,26]. De Carvalho showed that the relational model is injective for MELL [11]. In another direction, [4] provides a fully complete model for MALL without game semantics, by using a glueing construction on the model of hypercoherences. [21] explores proof nets a weaker theory of commuting conversions for MALL.

The idea of having intermediate representations between proof nets and proofs has been studied by Faggian and coauthors using l-nets [10,13–16], leading to a similar analysis to ours: they define a space of causal nets as partial orders and compare different versions of proofs with varying degree of parallelism. Our work recasts this idea using event structures and adds the notion of causal completeness: keeping jumps that cannot be undone by a permutation, which leads naturally to step outside partial orders, as well as full completeness: which causal nets can be strongly sequentialised?

The notion of dependency between logical rules has also been studied in [3] in the case of MLL. From a proof net R, they build a partial order $D_{\parr,\otimes}(R)$ which we believe is very related to $[\![P]\!]$ where P is a sequentialisation of R. Indeed, in the case of MLL *without MIX* a partial order is enough to capture the dependency between rules. The work [12] shows that permutation rules of Linear Logic, understood as asynchronous optimisations on processes, are included in the observational equivalence. [19] studies mutual embedding between polarised proof nets [23] and the control π-calculus [20]. In another direction, we have recently built a fully-abstract, concurrent game semantics model of the synchronous session π-calculus [8]. The difficulty there was to understand name passing and the synchrony of the π-calculus, which is the dual of our objective here: trying to understand the asynchrony behind the conversions of MALL$^-$.

Acknowledgements. We would like to thank Willem Heijltjes, Domenico Ruoppolo, and Olivier Laurent for helpful discussions, and the anonymous referees for their insightful comments. This work has been partially sponsored by: EPSRC EP/K034413/1, EP/K011715/1, EP/L00058X/1, EP/N027833/1, and EP/N028201/1.

References

1. Abramsky, S., Melliés, P.-A.: Concurrent games and full completeness. In: 14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, 2–5 July 1999, pp. 431–442 (1999). http://dx.doi.org/10.1109/LICS.1999.782638
2. Baelde, D., Doumane, A., Saurin, A.: Infinitary proof theory: the multiplicative additive case. In: CSL. Leibniz International Proceedings in Informatics (LIPIcs), vol. 62, pp. 42:1–42:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
3. Bagnol, M., Doumane, A., Saurin, A.: On the dependencies of logical rules. In: Pitts, A. (ed.) FoSSaCS 2015. LNCS, vol. 9034, pp. 436–450. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46678-0_28
4. Blute, R., Hamano, M., Scott, P.J.: Softness of hypercoherences and MALL full completeness. Ann. Pure Appl. Logic **131**(1–3), 1–63 (2005). https://doi.org/10.1016/j.apal.2004.05.002
5. Caires, L., Pfenning, F.: Session types as intuitionistic linear propositions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 222–236. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_16
6. Castellan, S., Clairambault, P., Winskel, G.: Observably deterministic concurrent strategies and intensional full abstraction for parallel-or. In: 2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, Oxford, UK, 3–9 September 2017, pp. 12:1–12:16 (2017). https://doi.org/10.4230/LIPIcs.FSCD.2017.12
7. Castellan, S., Yoshida, N.: Causality in linear logic: full completeness and injectivity (unit-free multiplicative-additive fragment). Technical report (2019). http://iso.mor.phis.me/publis/Causality_in_Linear_Logic_FOSSACS19.pdf
8. Castellan, S., Yoshida, N.: Two sides of the same coin: session types and game semantics. Accepted for publication at POPL 2019 (2019)
9. Chaudhuri, K., Miller, D., Saurin, A.: Canonical sequent proofs via multi-focusing. In: Ausiello, G., Karhumäki, J., Mauri, G., Ong, L. (eds.) TCS 2008. IIFIP, vol. 273, pp. 383–396. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-09680-3_26
10. Curien, P.-L., Faggian, C.: L-nets, strategies and proof-nets. In: Ong, L. (ed.) CSL 2005. LNCS, vol. 3634, pp. 167–183. Springer, Heidelberg (2005). https://doi.org/10.1007/11538363_13
11. de Carvalho, D.: The relational model is injective for multiplicative exponential linear logic. In: 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, Marseille, France, 29 August–1 September 2016, pp. 41:1–41:19 (2016). https://doi.org/10.4230/LIPIcs.CSL.2016.41
12. DeYoung, H., Caires, L., Pfenning, F., Toninho, B.: Cut reduction in linear logic as asynchronous session-typed communication. In: CSL, pp. 228–242 (2012)
13. Giamberardino, P.D.: Jump from parallel to sequential proofs: additives. Technical report (2011). https://hal.archives-ouvertes.fr/hal-00616386
14. Faggian, C., Maurel, F.: Ludics nets, a game model of concurrent interaction. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), Chicago, IL, USA, 26–29 June 2005, Proceedings, pp. 376–385. IEEE Computer Society (2005). http://dx.doi.org/10.1109/LICS.2005.25
15. Faggian, C., Piccolo, M.: A graph abstract machine describing event structure composition. Electr. Notes Theor. Comput. Sci. **175**(4), 21–36 (2007). https://doi.org/10.1016/j.entcs.2007.04.014

16. Di Giamberardino, P., Faggian, C.: Jump from parallel to sequential proofs: multiplicatives. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 319–333. Springer, Heidelberg (2006). https://doi.org/10.1007/11874683_21
17. Girard, J.Y.: Linear logic. Theor. Comput. Sci. **50**(1), 1–101 (1987)
18. Heijltjes, W., Houston, R.: No proof nets for MLL with units: proof equivalence in MLL is PSPACE-complete. In: CSL-LICS 2014, pp. 50:1–50:10. ACM (2014)
19. Honda, K., Laurent, O.: An exact correspondence between a typed pi-calculus and polarised proof-nets. Theor. Comput. Sci. **411**(22–24), 2223–2238 (2010). https://doi.org/10.1016/j.tcs.2010.01.028
20. Honda, K., Yoshida, N., Berger, M.: Process types as a descriptive tool for interaction. In: Dowek, G. (ed.) RTA 2014. LNCS, vol. 8560, pp. 1–20. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08918-8_1
21. Hughes, D.J.D., Heijltjes, W.: Conflict nets: efficient locally canonical MALL proof nets. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, New York, NY, USA, 5–8 July 2016, pp. 437–446 (2016). http://doi.acm.org/10.1145/2933575.2934559
22. Hughes, D.J.D., van Glabbeek, R.J.: Proof nets for unit-free multiplicative-additive linear logic. ACM Trans. Comput. Logic **6**(4), 784–842 (2005)
23. Laurent, O.: Polarized proof-nets and $\lambda\mu$-calculus. Theor. Comput. Sci. **290**(1), 161–188 (2003). https://doi.org/10.1016/S0304-3975(01)00297-3
24. Melliès, P.-A.: Asynchronous games 4: a fully complete model of propositional linear logic. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), Chicago, IL, USA, 26–29 June 2005, Proceedings, pp. 386–395 (2005). http://dx.doi.org/10.1109/LICS.2005.6
25. Melliès, P.-A., Mimram, S.: Asynchronous games: innocence without alternation. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 395–411. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74407-8_27
26. Mimram, S.: Sémantique des jeux asynchrones et réécriture 2-dimensionnelle (asynchronous game semantics and 2-dimensional rewriting systems). Ph.D. thesis, Paris Diderot University, France (2008). https://tel.archives-ouvertes.fr/tel-00338643
27. Wadler, P.: Propositions as sessions. J. Funct. Program. **24**(2–3), 384–418 (2014)
28. Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) ACPN 1986. LNCS, vol. 255, pp. 325–392. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-17906-2_31